



Instituto Politécnico Nacional

Escuela Superior de Cómputo



Desarrollo de Sistemas Distribuidos.

Tarea 1. Sistema distribuido que verifica si un número es primo.

Nombre: Flores Castro Luis Antonio.

Profesor: Pineda Guerrero Carlos.

Grupo: 4CV13.

14 Marzo 2023

Funcionamiento de programas.

Para la ejecución correcta de los programas, primero se inician los servidores y después el cliente.

1. Primero se muestra la compilación y ejecución del cliente, al ejecutarse se despliega un mensaje donde nos indica que debemos ingresar un número para validar si este es primo o no.

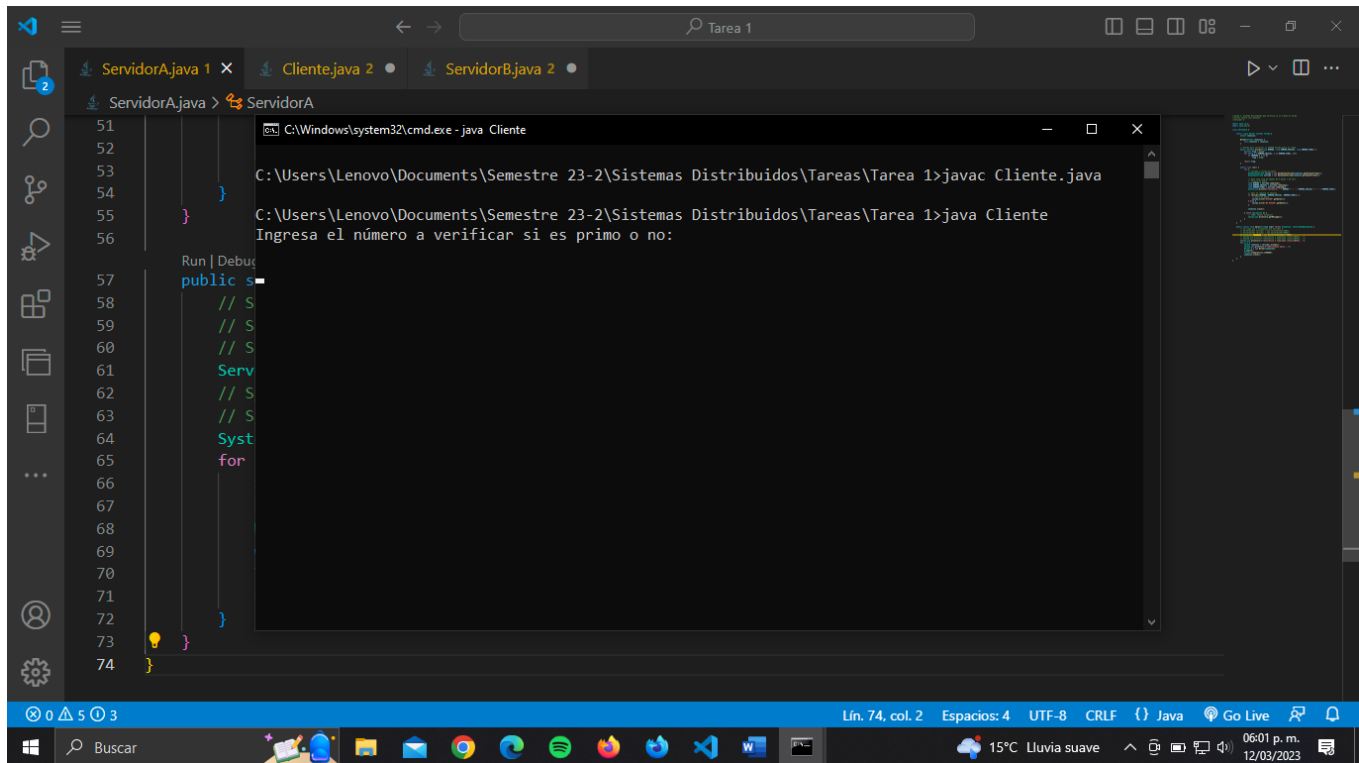
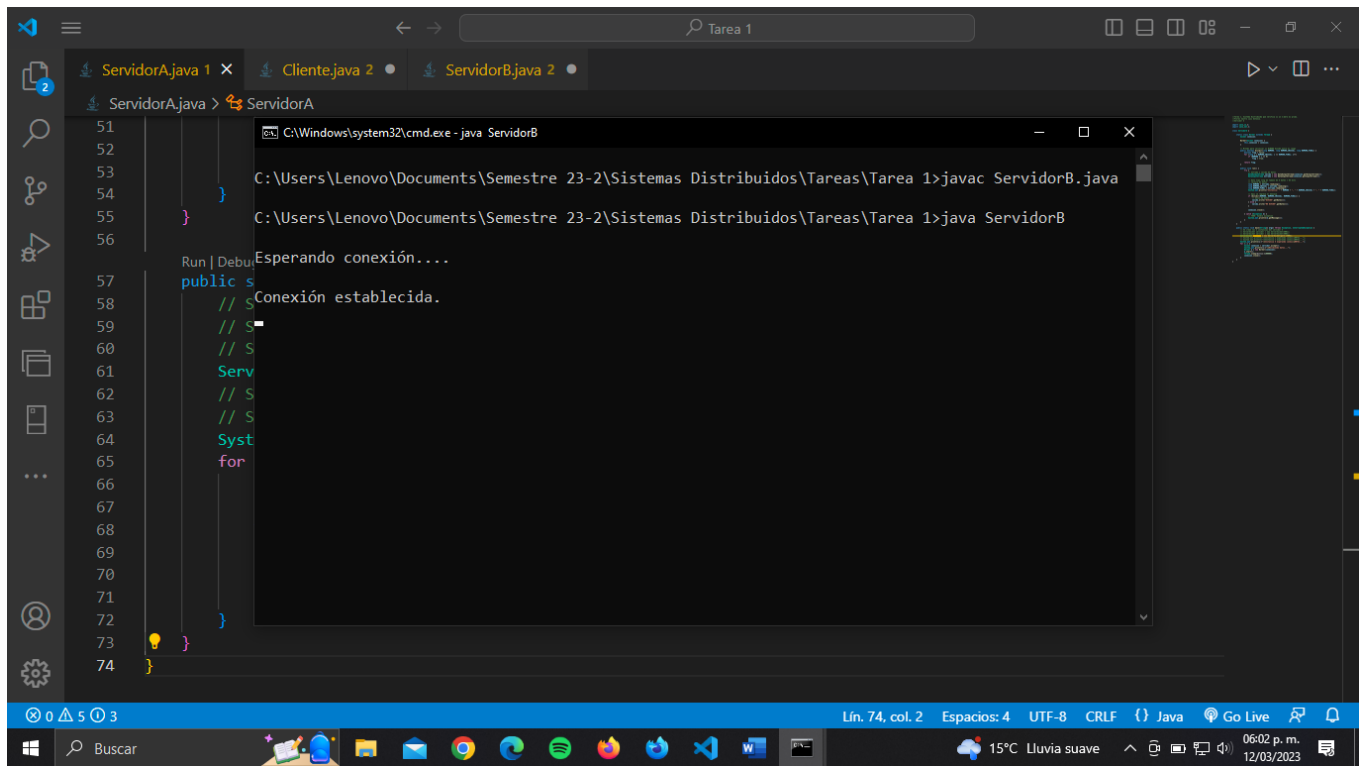


Figura 1. Compilación y ejecución del cliente.

2. En la figura 2 se muestra la compilación y ejecución del servidor B. Se puede observar que el programa despliega un mensaje donde indica que se está esperando por una conexión. Primero se inician los servidores y después el cliente. Por lo tanto, cuando se ejecuta el cliente, el servidor B nos muestra el mensaje de conexión establecida.



```
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74
```

```
C:\Windows\system32\cmd.exe - java ServidorB  
C:\Users\Lenovo\Documents\Semestre 23-2\Sistemas Distribuidos\Tareas\Tarea 1>javac ServidorB.java  
C:\Users\Lenovo\Documents\Semestre 23-2\Sistemas Distribuidos\Tareas\Tarea 1>java ServidorB  
Esperando conexión...  
Conexión establecida.
```

Figura 2. Compilación y ejecución del servidor B.

3. A continuación, se comienzan a compilar y ejecutar las diferentes instancias del servidor A, en la figura 3 se puede ver la primera instancia utilizando el puerto número 5001 la cual está esperando a recibir los datos por parte del servidor B.

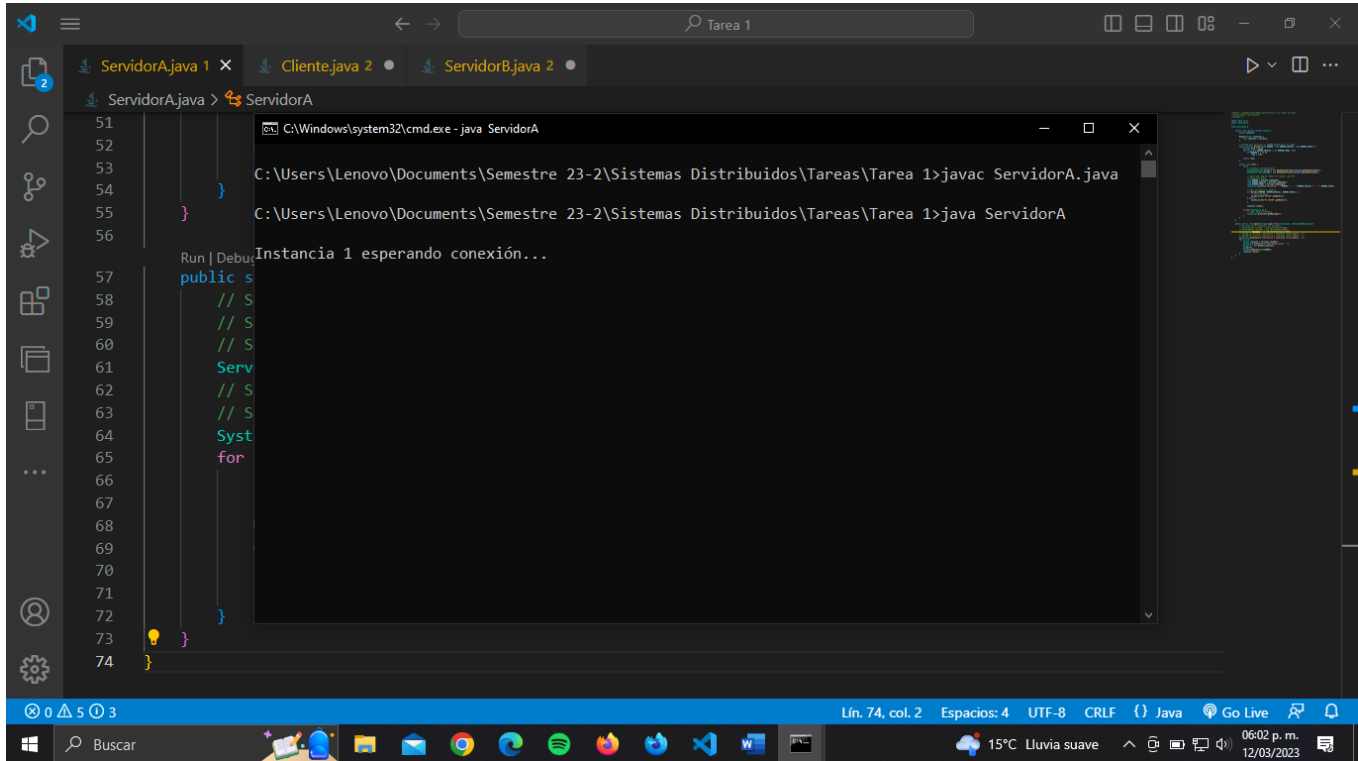
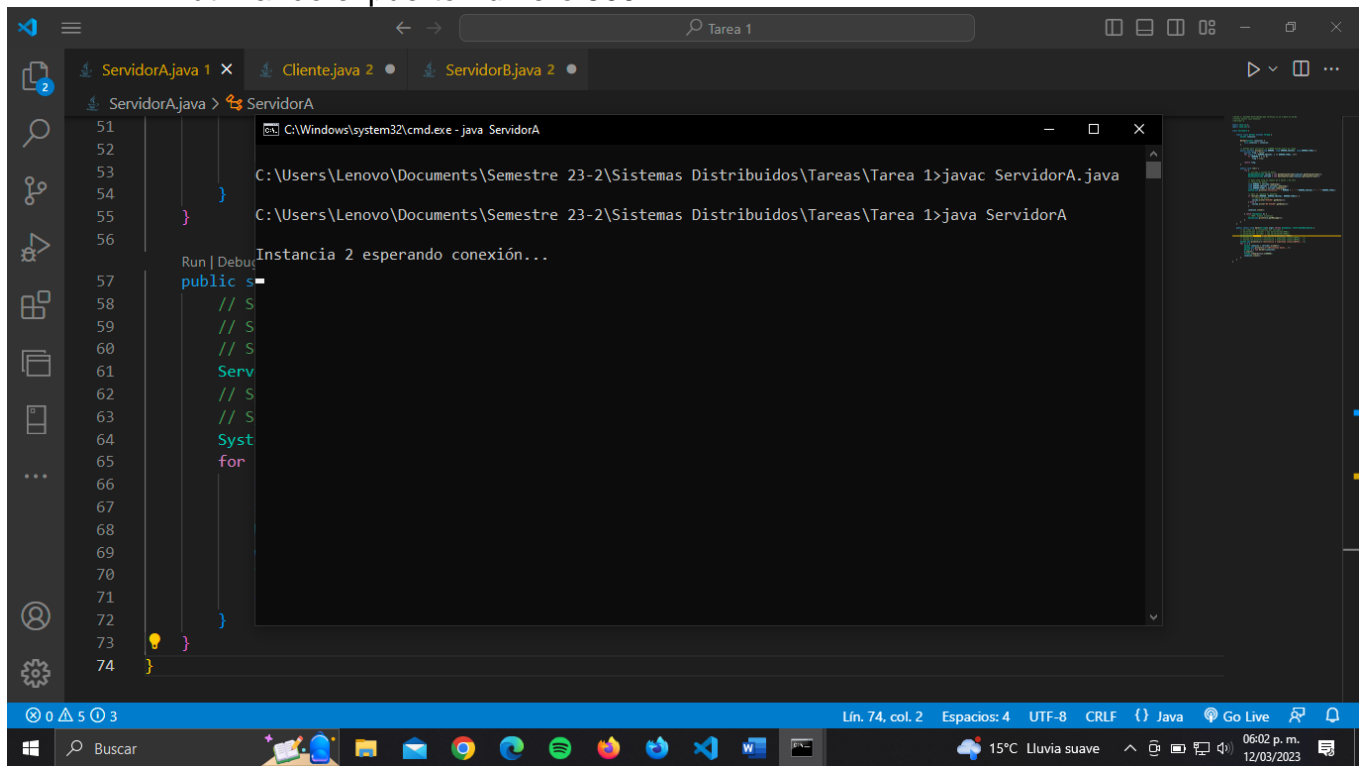


Figura 3. Compilación y ejecución de primera instancia del servidor A utilizando el puerto 5001.

4. Compilación y ejecución de la segunda instancia del servidor A, ahora utilizando el puerto número 5002.



The screenshot shows an IDE with three tabs: 'ServidorA.java 1', 'Cliente.java 2', and 'ServidorB.java 2'. The 'ServidorA.java 1' tab is active, showing a Java file with line numbers 51 to 74. The code includes a 'public static void main' method. A terminal window is open, showing the command 'C:\Windows\system32\cmd.exe - java ServidorA' and the output 'C:\Users\Lenovo\Documents\Semestre 23-2\Sistemas Distribuidos\Tareas\Tarea 1>javac ServidorA.java' and 'C:\Users\Lenovo\Documents\Semestre 23-2\Sistemas Distribuidos\Tareas\Tarea 1>java ServidorA'. The output also shows 'Instancia 2 esperando conexión...'. The IDE's status bar at the bottom indicates 'Lín. 74, col. 2', 'Espacios: 4', 'UTF-8', 'CRLF', 'Java', and 'Go Live'. The Windows taskbar at the bottom shows the time as 06:02 p. m. on 12/03/2023.

```
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74
```

```
public static void main  
    // S  
    // S  
    // S  
    Serv  
    // S  
    // S  
    Syst  
    for
```

```
C:\Windows\system32\cmd.exe - java ServidorA  
C:\Users\Lenovo\Documents\Semestre 23-2\Sistemas Distribuidos\Tareas\Tarea 1>javac ServidorA.java  
C:\Users\Lenovo\Documents\Semestre 23-2\Sistemas Distribuidos\Tareas\Tarea 1>java ServidorA  
Instancia 2 esperando conexión...
```

Lín. 74, col. 2 Espacios: 4 UTF-8 CRLF Java Go Live

06:02 p. m.
12/03/2023

Figura 4. Compilación y ejecución de segunda instancia del servidor A utilizando el puerto 5002.

5. Compilación y ejecución de la tercera instancia del servidor A utilizando el puerto 5003.

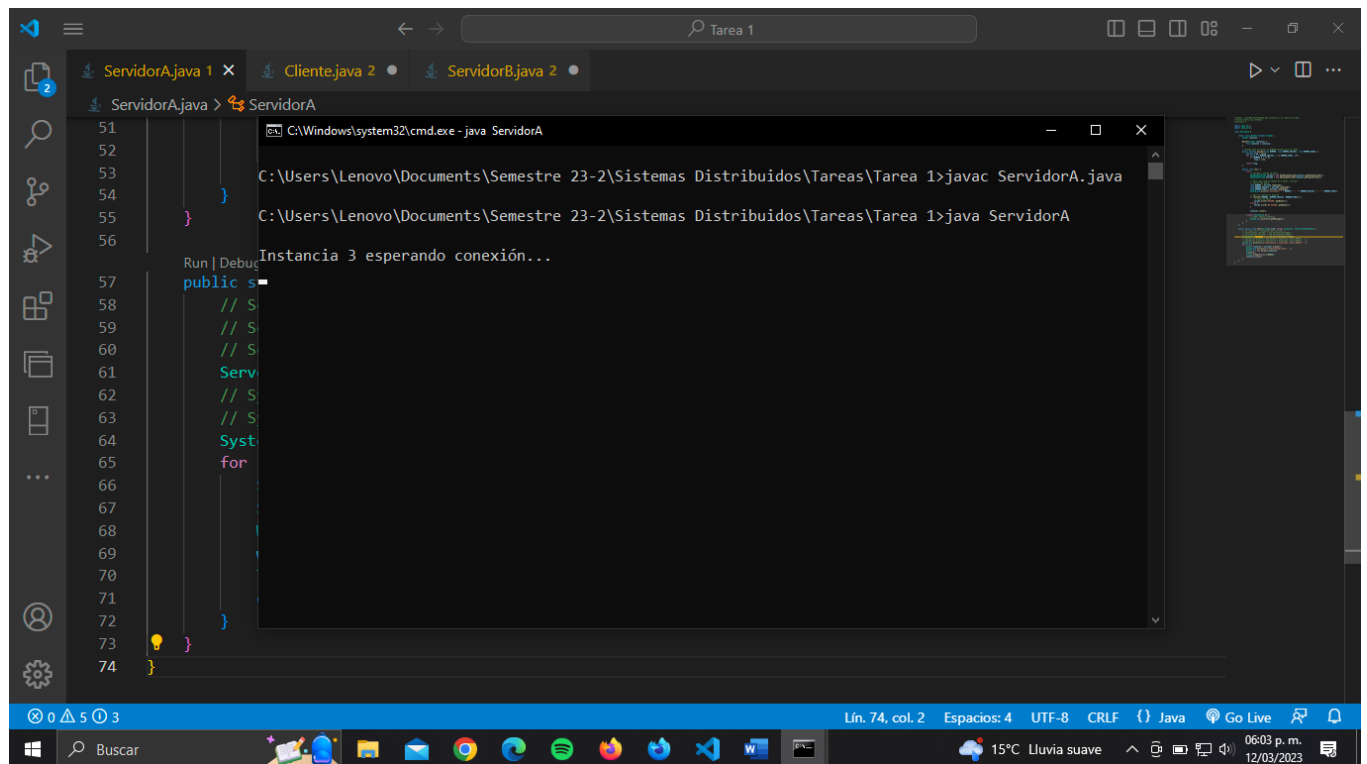


Figura 5. Compilación y ejecución de tercera instancia del servidor A utilizando el puerto 5003.

6. Ya compilados y ejecutados tanto servidores como cliente, ingresamos el número 1234567811 en el cliente para verificar si en efecto es primo.

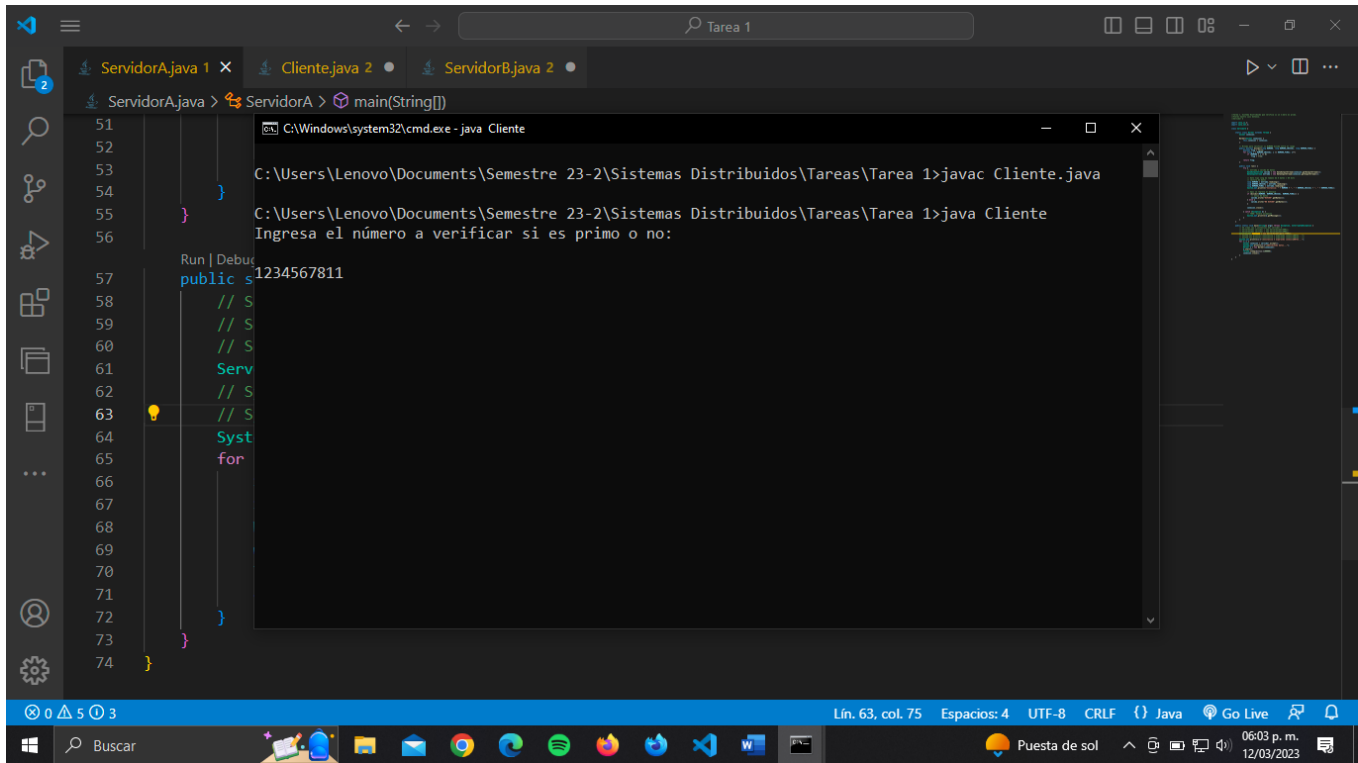
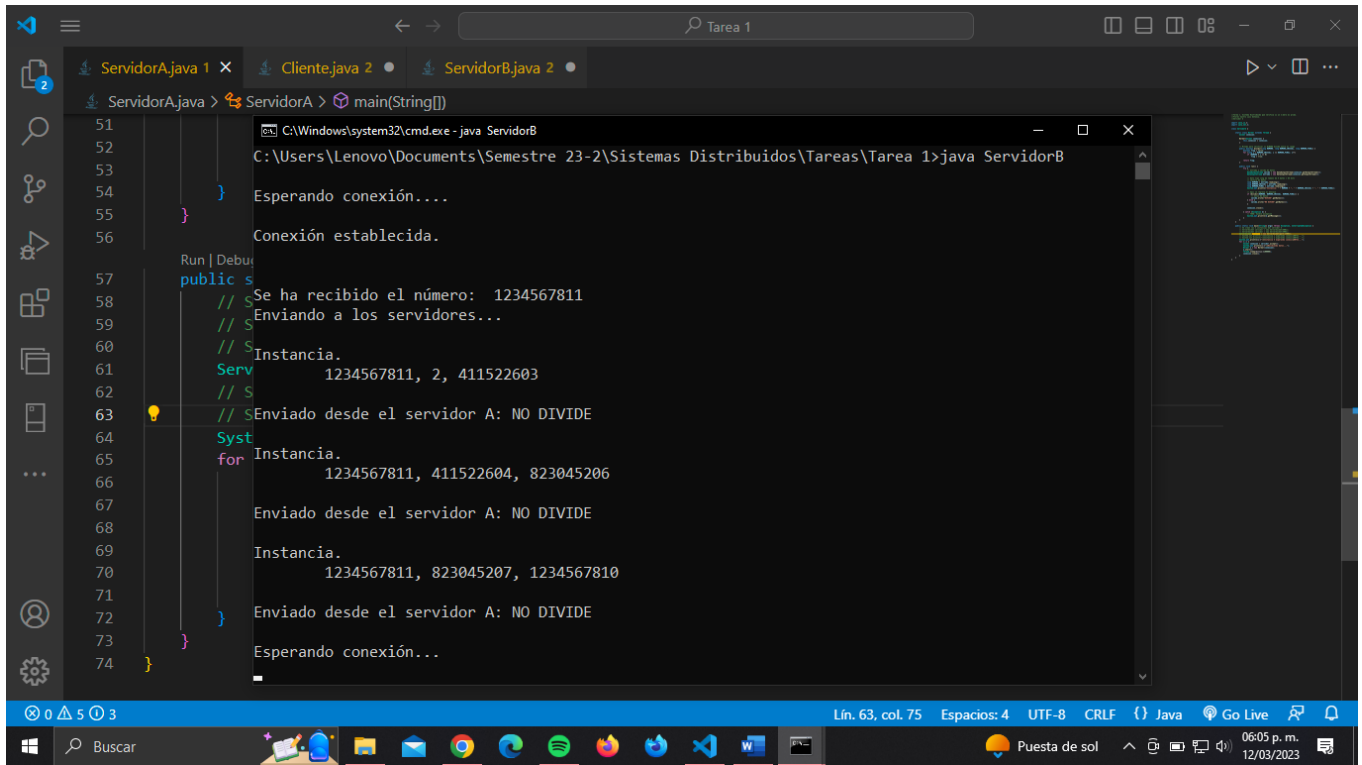


Figura 6. Ingreso de datos al programa cliente.

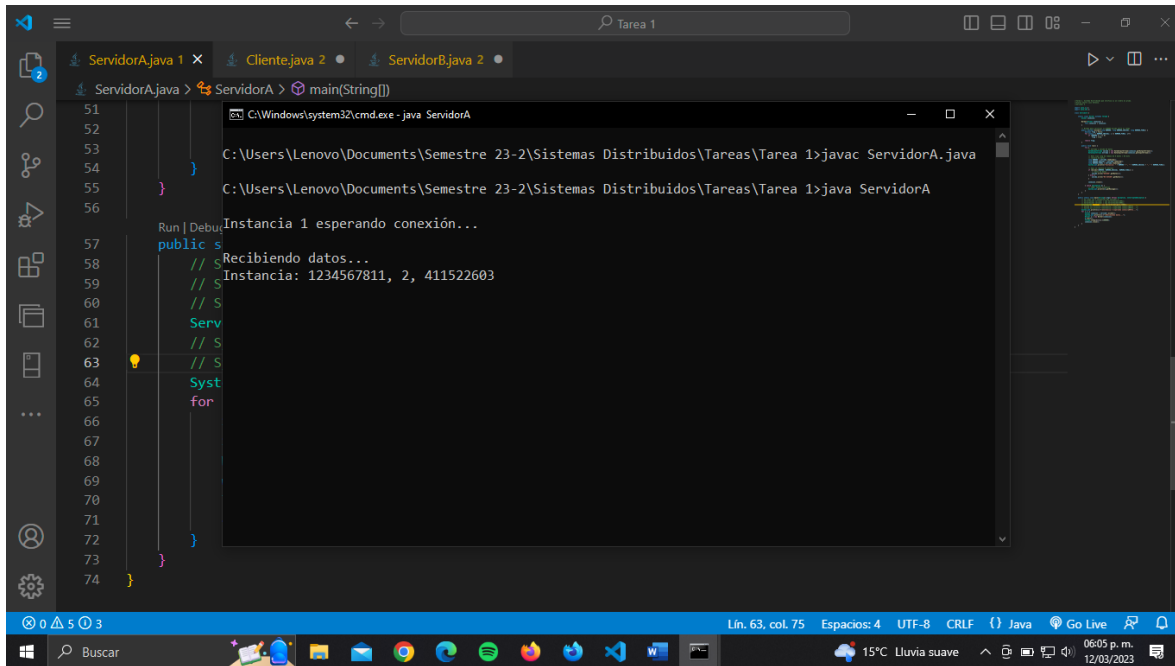
7. Posteriormente de haber ingresado el número al programa cliente, podemos ver que el servidor B recibió el número por parte del cliente, ahora el servidor B opera el número recibido con el fin de mandar a las tres instancias del servidor A los intervalos, a su vez el servidor B despliega las respuestas enviadas por las 3 instancias del servidor A.



```
51  C:\Windows\system32\cmd.exe - java ServidorB
52  C:\Users\Lenovo\Documents\Semestre 23-2\Sistemas Distribuidos\Tareas\Tarea 1>java ServidorB
53
54  } Esperando conexión....
55
56  } Conexión establecida.
57  Run | Debug
58  public s Se ha recibido el número: 1234567811
59  // s Enviando a los servidores...
60  // s
61  // s Instancia.
62  // s Serv 1234567811, 2, 411522603
63  // s Enviado desde el servidor A: NO DIVIDE
64  // s
65  // s Instancia.
66  // s Syst for 1234567811, 411522604, 823045206
67  Enviado desde el servidor A: NO DIVIDE
68
69  Instancia.
70  1234567811, 823045207, 1234567810
71  Enviado desde el servidor A: NO DIVIDE
72
73  } Esperando conexión...
74  }
```

Figura 7. Despliegue de mensajes recibidos por parte de los programas cliente e instancias del servidor A en el servidor B.

8. En la siguiente imagen se muestra la primera instancia del servidor A, donde nos muestra el primer intervalo que ha recibido por parte del servidor B, después de haber recibido el intervalo realiza las operaciones correspondientes para enviar una respuesta al servidor B y que este decida si en efecto es un número primo.

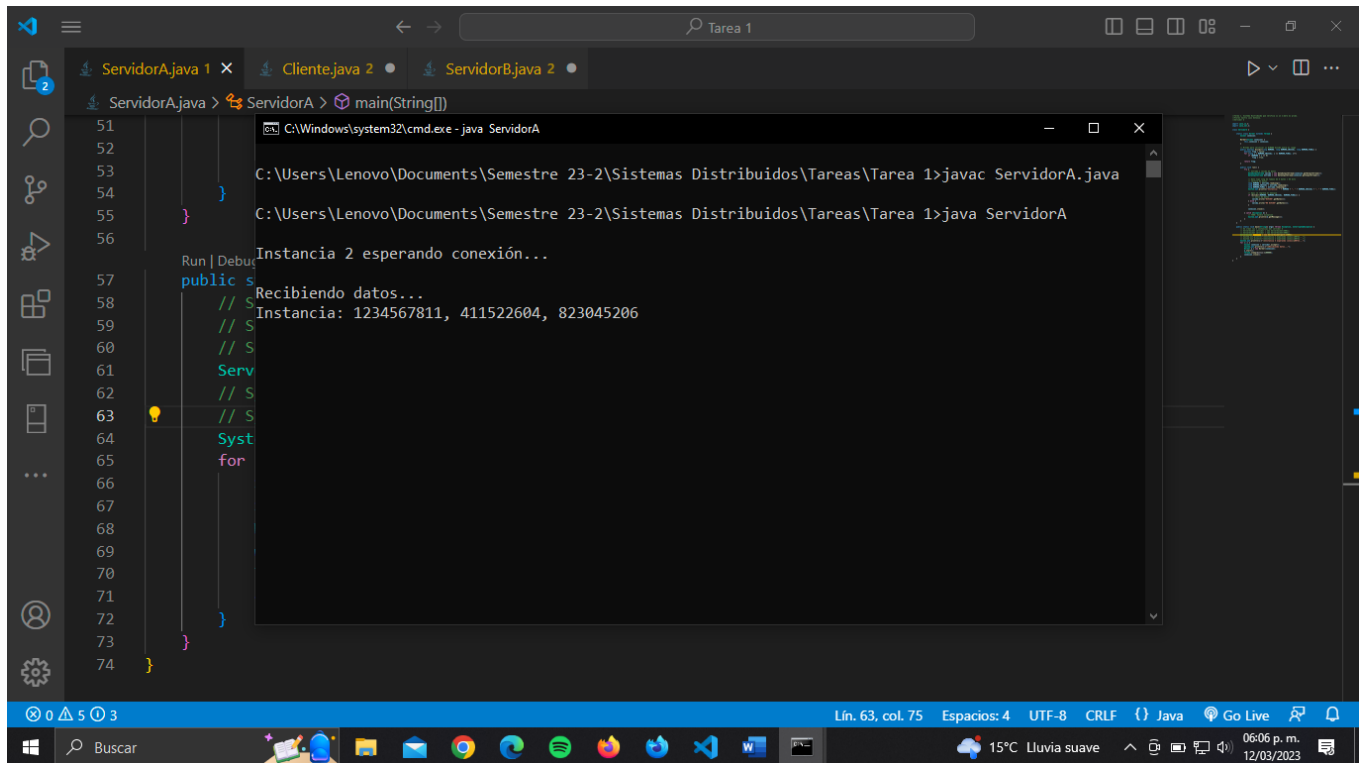


```
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74
```

```
C:\Windows\system32\cmd.exe - java ServidorA  
C:\Users\Lenovo\Documents\Semestre 23-2\Sistemas Distribuidos\Tareas\Tarea 1>javac ServidorA.java  
C:\Users\Lenovo\Documents\Semestre 23-2\Sistemas Distribuidos\Tareas\Tarea 1>java ServidorA  
Instancia 1 esperando conexión...  
Recibiendo datos...  
Instancia: 1234567811, 2, 411522603
```

Figura 8. Despliegue de intervalo recibido en primera instancia del servidor A.

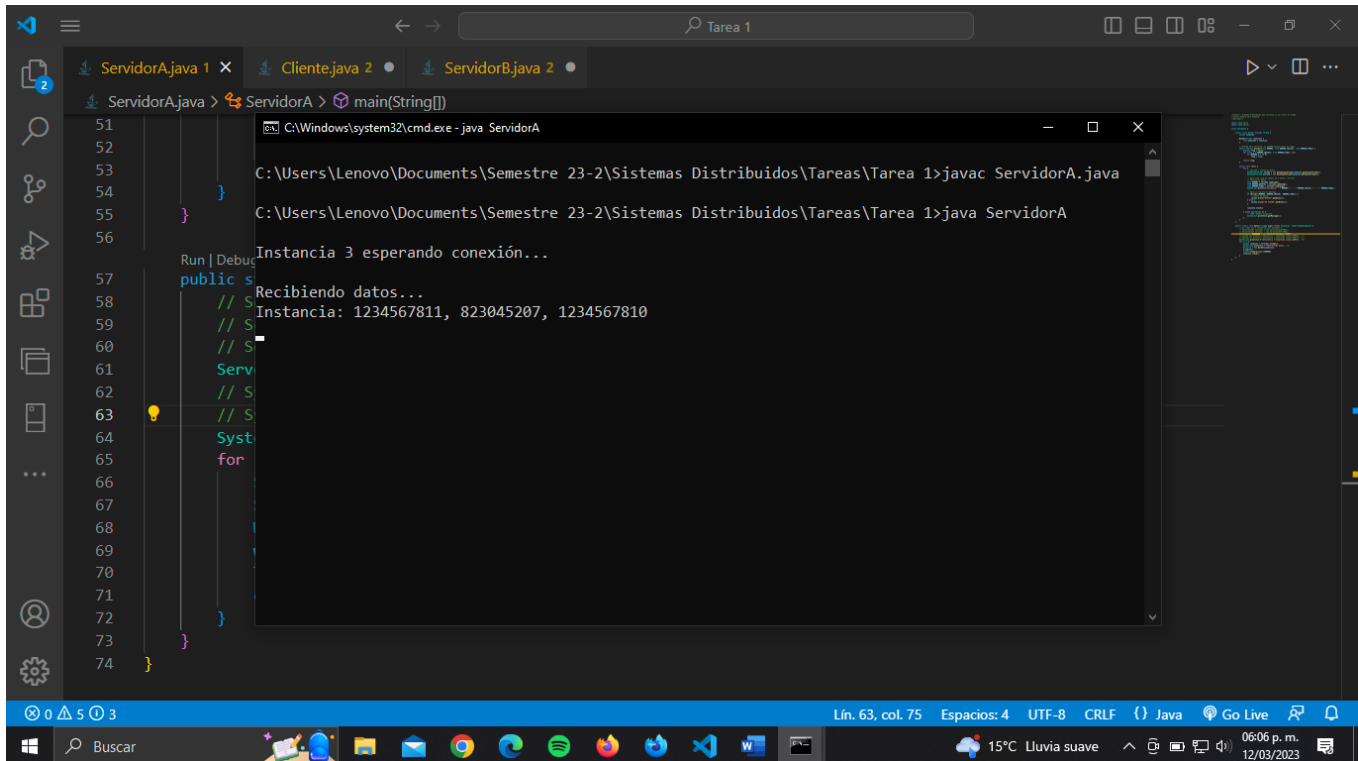
9. Ahora podemos ver que en la segunda instancia del servidor A se muestra el intervalo que ha sido recibido.



The screenshot shows an IDE with three tabs: `ServidorA.java 1`, `Cliente.java 2`, and `ServidorB.java 2`. The `ServidorA.java` tab is active, showing a Java program with line numbers 51 to 74. The code includes a `main` method that calls `main(String[])`. A console window is open, showing the output of the program. The output includes the command `C:\Windows\system32\cmd.exe - java ServidorA`, the command `C:\Users\Lenovo\Documents\Semestre 23-2\Sistemas Distribuidos\Tareas\Tarea 1>javac ServidorA.java`, the command `C:\Users\Lenovo\Documents\Semestre 23-2\Sistemas Distribuidos\Tareas\Tarea 1>java ServidorA`, and the output `Instancia 2 esperando conexión...`, `Recibiendo datos...`, and `Instancia: 1234567811, 411522604, 823045206`. The IDE status bar at the bottom shows `Lín. 63, col. 75`, `Espacios: 4`, `UTF-8`, `CRLF`, `() Java`, and `Go Live`. The Windows taskbar at the bottom shows the search bar, task view, and various application icons, including Chrome, Firefox, and VS Code. The system tray shows the date and time as `06:06 p. m. 12/03/2023`.

Figura 9. Despliegue de intervalo recibido en segunda instancia del servidor A.

10. Finalmente, tenemos la tercera instancia del servidor A mostrándonos el último intervalo recibido.

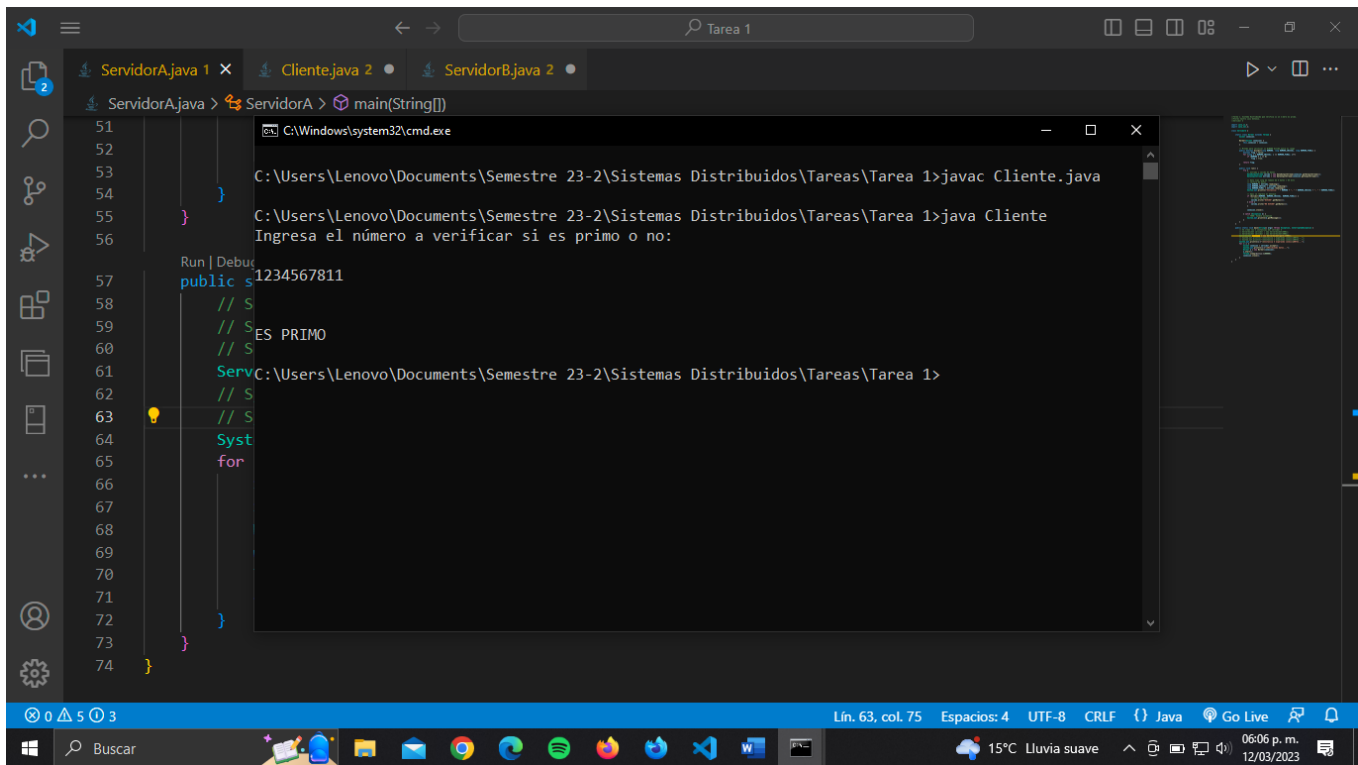


```
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74
```

```
C:\Windows\system32\cmd.exe - java ServidorA  
C:\Users\Lenovo\Documents\Semestre 23-2\Sistemas Distribuidos\Tareas\Tarea 1>javac ServidorA.java  
C:\Users\Lenovo\Documents\Semestre 23-2\Sistemas Distribuidos\Tareas\Tarea 1>java ServidorA  
Instancia 3 esperando conexión...  
Recibiendo datos...  
Instancia: 1234567811, 823045207, 1234567810
```

Figura 10. Despliegue de intervalo recibido en tercera instancia del servidor A.

11. Por último, podemos ver que el cliente recibe el mensaje por parte del servidor B que indica si el número que se ingreso es primo o no, para el caso del número 1234567811 podemos comprobar que en efecto es primo.



```
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74
```

```
C:\Windows\system32\cmd.exe  
C:\Users\Lenovo\Documents\Semestre 23-2\Sistemas Distribuidos\Tareas\Tarea 1>javac Cliente.java  
C:\Users\Lenovo\Documents\Semestre 23-2\Sistemas Distribuidos\Tareas\Tarea 1>java Cliente  
Ingresa el número a verificar si es primo o no:  
1234567811  
// S  
// S  
ES PRIMO  
Serv C:\Users\Lenovo\Documents\Semestre 23-2\Sistemas Distribuidos\Tareas\Tarea 1>  
// S  
// S  
Syst  
for
```

Figura 11. Despliegue de mensaje recibido por el cliente donde indica si el número ingresado es primo o no.

Código servidor A.

```
//Tarea 1. Sistema Distribuido que verifica si un número es primo.
//Flores Castro Luis Antonio.
//Servidor A

import java.io.*;
import java.net.*;

class ServidorA {

    static class Worker extends Thread {
        Socket conexion;

        Worker(Socket conexion) {
            this.conexion = conexion;
        }

        // Método para verificar si NÚMERO divide entre el rango
        public boolean dividir(long NUMERO, long NUMERO_INICIAL, long
NUMERO_FINAL) {
            boolean flag = false;
            for (long i = NUMERO_INICIAL; i <= NUMERO_FINAL; i++)
                if (NUMERO % i == 0)
                    flag = true;

            return flag;
        }

        public void run() {
            try {
                // entrada y salida de datos
                DataOutputStream salida = new
DataOutputStream(conexion.getOutputStream());
                DataInputStream entrada = new
DataInputStream(conexion.getInputStream());

                // Dato tipo long de tamaño de 8 bytes = 64 bits
                // lectura de datos
                long NUMERO = entrada.readLong();
                long NUMERO_INICIAL = entrada.readLong();
                long NUMERO_FINAL = entrada.readLong();
                System.out.println("Instancia: " + NUMERO + ", " +
NUMERO_INICIAL + ", " + NUMERO_FINAL);

                // Aquí se regresa el mensaje
```

```

        if (dividir(NUMERO, NUMERO_INICIAL, NUMERO_FINAL)) {
            // envio de datos
            salida.write("DIVIDE".getBytes());
        } else {
            salida.write("NO DIVIDE".getBytes());
        }

        conexion.close();

    } catch (Exception e) {
        // TODO: handle exception
        System.out.println(e.getMessage());
    }
}

public static void main(String[] args) throws Exception,
InterruptedException {
    // Se crean las 3 instancias del servidor A
    ServerSocket servidor = new ServerSocket(5001);
    // ServerSocket servidor = new ServerSocket(5002);
    // ServerSocket servidor = new ServerSocket(5003);
    for (;;) {
        Socket conexion = servidor.accept();
        Worker w = new Worker(conexion);
        w.start();
        Thread.sleep(100000);
        conexion.close();
    }
}
}

```

Código servidor B.

```
//Tarea 1. Sistema Distribuido que verifica si un número es primo.
//Flores Castro Luis Antonio.
//Servidor B

import java.io.*;
import java.net.*;
import javax.sound.sampled.Port;

class ServidorB {

    // Declaracion de objetos y variables estaticas para el uso de
    synchronized
    static Object obj = new Object();
    static int deteccion;

    static class Worker extends Thread {
        Socket conexion;
        long NUMERO;
        long NUMERO_INICIAL;
        long NUMERO_FINAL;

        Worker(Socket conexion, long NUMERO, long NUMERO_INICIAL, long
NUMERO_FINAL) {
            this.conexion = conexion;
            this.NUMERO = NUMERO;
            this.NUMERO_INICIAL = NUMERO_INICIAL;
            this.NUMERO_FINAL = NUMERO_FINAL;
        }

        public void run() {
            try {
                // entrada y salida de datos
                DataOutputStream salida = new
DataOutputStream(conexion.getOutputStream());
                DataInputStream entrada = new
DataInputStream(conexion.getInputStream());

                System.out.println("\nInstancia.");
                // Primero se envian los datos numéricos a los servidores
                System.out.println("\t" + NUMERO + ", " + NUMERO_INICIAL +
", " + NUMERO_FINAL);
                salida.writeLong(NUMERO);
                salida.writeLong(NUMERO_INICIAL);
```

```

        salida.writeLong(NUMERO_FINAL);
        // Se realiza la sincronización de los hilos para que no
existe error en los
        // resultados
        synchronized (obj) {
            byte[] buffer = new byte[9];
            // Aqui se puede llamar a la función hecha por nosotros
o el método
            // perteneciente a la clase InputStream
            entrada.read(buffer, 0, 9);
            String respuesta = new String(buffer, "UTF-8");
            if (respuesta.startsWith("NO")) {
                // Entonces es primo y la variable deteccion debe de
igualarse a 3 que es el
                // número de instancias de SA
                System.out.println("\nEnviado desde el servidor A: "
+ respuesta);
                deteccion++;
            } else {
                // No es primo
                System.out.println("\nEnviado desde el servidor A: "
+ respuesta);
            }
        }

        // Cerrar la conexion
        conexion.close();

    } catch (Exception e) {
        // TODO: handle exception
        System.out.println(e.getMessage());
    }
}

}

public static void main(String[] args) throws Exception {
    ServerSocket servidor = new ServerSocket(5000);
    System.out.println("\nEsperando conexi\u00f3n....");
    for (;;) {
        Socket conexion = servidor.accept();
        System.out.println("\nConexi\u00f3n establecida.");
        // Se realiza la lectura de las respuestas obtenidas por Server

```

A


```

        DataOutputStream salida = new
DataOutputStream(conexion.getOutputStream());
        DataInputStream entrada = new
DataInputStream(conexion.getInputStream());
        long k;
        // Primero se lee el número que manda el cliente
        long NUMERO = entrada.readLong();
        k = NUMERO / 3;
        System.out.println("\n\nSe ha recibido el número: " +
NUMERO);
        System.out.println("Enviando a los servidores...");

        // Se crean las instancias de conexion para los servidores A
        Socket conexionS1 = new Socket("localhost", 5001);
        Socket conexionS2 = new Socket("localhost", 5002);
        Socket conexionS3 = new Socket("localhost", 5003);

        // Ahora se pasan los números a enviar mediante el constructor
Worker
        Worker serA1 = new Worker(conexionS1, NUMERO, 2, k);
        Worker serA2 = new Worker(conexionS2, NUMERO, k + 1, 2 * k);
        Worker serA3 = new Worker(conexionS3, NUMERO, 2 * k + 1, NUMERO
- 1);

        // Se inician los hilos y las barreras
        serA1.start();
        serA1.join();
        serA2.start();
        serA2.join();
        serA3.start();
        serA3.join();

        if (deteccion == 3)
            // System.out.println("ES PRIMO");
            // Aqui se envia el string al cliente
            salida.write("ES PRIMO".getBytes());
        if (deteccion < 3)
            // System.out.println("NO ES PRIMO");
            salida.write("NO ES PRIMO".getBytes());
        // reiniciamos el indicador
        deteccion = 0;

        System.out.println("\nEsperando conexi\u00f3n...");

        // Quitar en caso de que afecte funcionamiento

```

```
        connexionS1.close();  
        connexionS2.close();  
        connexionS3.close();  
        connexion.close();  
    }  
}  
}
```

Conclusiones.

Mediante la realización de esta práctica pude ver la aplicación un poco más compleja de ciertos temas que abordamos a lo largo de estas semanas. En la codificación del servidor A, puedo decir que se me facilitó ya que prácticamente consistía en realizar lectura y escritura de datos, por lo que no tuve mayores problemas. Pasando a la parte del servidor B, encontré ciertas dificultades debido a la sincronización de los hilos. El principal error que tenía era que las conexiones en ocasiones se cerraban antes o los resultados no eran los esperados, por lo tanto, tuve que ajustar parte de la sincronización y el tiempo en milisegundos tanto en el servidor A como en el B, todo esto para que la ejecución se diera de manera exitosa. En la parte del cliente, de igual forma que en el servidor A, no tuve inconvenientes al codificarlo. Considerando todos los puntos mencionados, puedo decir que dicha práctica me permitió abordar de mejor forma todos los temas que habíamos visto. De igual manera, pude ver una aplicación mucho más amplia y darme ideas de lo que se podría llegar a crear con conceptos básicos, por decirlo de alguna manera.